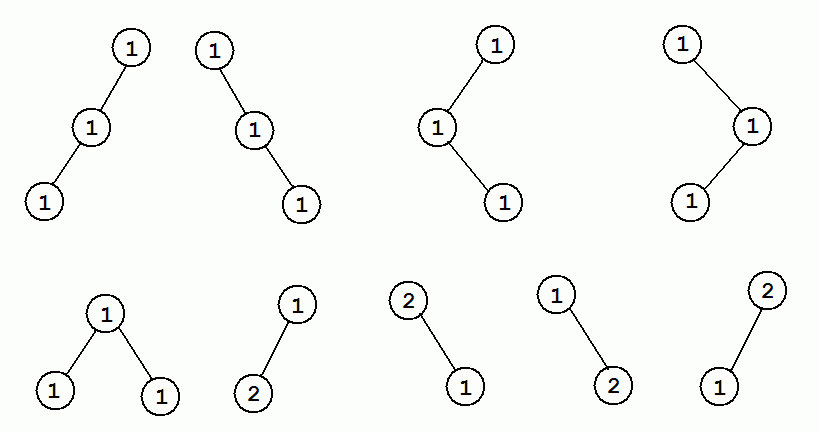
# bzoj3625(NTT+多项式求逆+多项式开根）

# **[bzoj3625-小朋友和二叉树](https://www.cnblogs.com/owenyu/p/6724613.html)**

# **[题目](http://www.lydsy.com/JudgeOnline/problem.php?id=3625)**

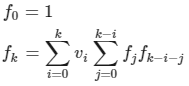
给出n个互异的正整数序列，c1,c2...cn。定义一颗带点权树的权值为所有点的权值和。给出一个整数m，请计算对于所有的i=1,2,3...,m，用c中的数能组成多少不同的二叉树（每个数可以重复使用）且二叉树的权值为i。答案对7∗17∗223+1=998244353取模。

## **Sample Input**

对于n=2，c={1,2}，i=3，有九个这样的二叉树：  


# **分析**

其实看到那个模数就可以想到要用到NTT了。  
可以说第一次遇到这种题，但这种生成函数法的思想以前有接触过。这题很简单。  
两个二叉树不同就是不全等，所以我们可以分成三部分，当前点，左子树和右子树。我们设一个函数f，fi表示一个子树的点权和为i的情况数。再设vi为权值i的个数，即有或没有。那么有：



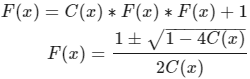
这个式子其实可以用一个生成函数来表示。生成函数是指一种关于另一个函数g的形式函数，它的xi项的系数为g(i)，可以这样表示：



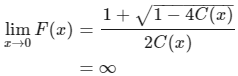
我们设F为f的生成函数，C为v的生成函数，那么有：



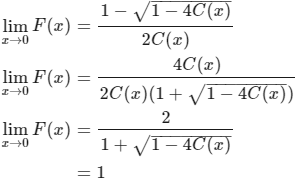
那个1是从f0=1来的。  
我们要求的是F的1−m项的系数，所以要把F解出来。



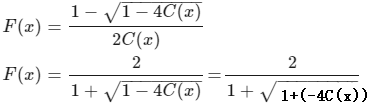
通过F(0)=1判断取值，若取正号：



矛盾，所以不可取。若取负号：

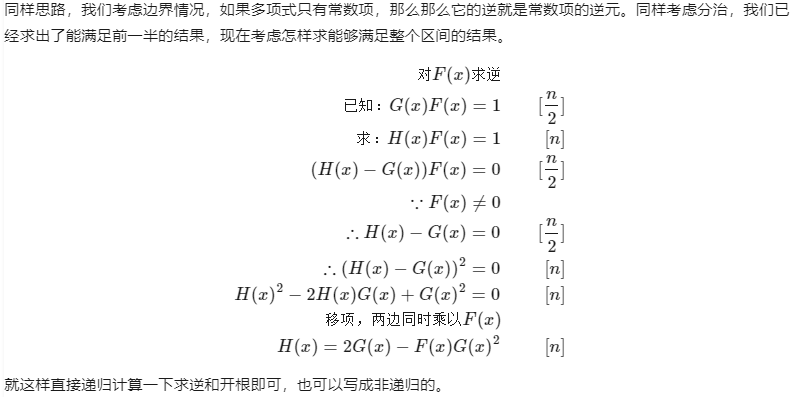


所以我们要计算的F就是：



这里涉及到了多项式模意义下的求逆与开根。

## **多项式求逆**



#include <iostream>

#include <fstream>

#include <algorithm>

#include <cmath>

#include <ctime>

#include <cstdio>

#include <cstdlib>

#include <cstring>

using namespace std;

#define mmst(a, b) memset(a, b, sizeof(a))

#define mmcp(a, b) memcpy(a, b, sizeof(b))

typedef long long LL;

const int p=998244353,I2=499122177;

const int N=800400;

int cheng(int a,int b)

{

int res=1;

for(;b;b>>=1,a=(LL)a\*a%p)

if(b&1)

res=(LL)res\*a%p;

return res;

}

int n,rev[N];

void init(int lim)

{

n=1;

int k=-1;

while(n<lim)

n<<=1,k++;

for(int i=0;i<n;i++)

rev[i]=(rev[i>>1] >> 1) | ((i&1)<<k);

}

void ntt(int \*a,int ops)

{

for(int i=0;i<n;i++)

if(i<rev[i])

swap(a[i],a[rev[i]]);

for(int l=2;l<=n;l<<=1)

{

int m=l>>1,wn;

if(ops)

wn=cheng(3,(p-1)/l);

else

wn=cheng(3,p-1-(p-1)/l);

for(int i=0;i<n;i+=l)

{

int w=1;

for(int k=0;k<m;k++)

{

int t=(LL)a[i+k+m]\*w%p;

a[i+k+m]=(a[i+k]-t+p)%p;

a[i+k]=(a[i+k]+t)%p;

w=(LL)w\*wn%p;

}

}

}

if(!ops)

{

int Inv=cheng(n,p-2);

for(int i=0;i<n;i++)

a[i]=(LL)a[i]\*Inv%p;

}

}

int g[N];

int mx=1,by,nn,mm;

int X[N],Y[N],sqr[N],A[N],B[N],C[N];

void Inverse(int \*a,int \*b,LL len)

{

if(len==1)

{

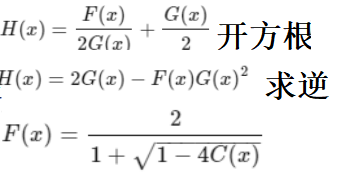
b[0]=cheng(a[0],p-2);

return;

}

Inverse(a,b,len>>1);

init(2\*len);

 for(int i=0;i<len;i++)

X[i]=a[i];

for(int i=0;i<(len>>1);i++)

Y[i]=b[i];

ntt(X,1);

ntt(Y,1);

for(int i=0;i<n;i++)

X[i]=(2ll\*Y[i]%p-(LL)X[i]\*Y[i]%p\*Y[i]%p+p)%p;

ntt(X,0);

for(int i=0;i<n;i++)

{

if(i>=len)

b[i]=0;

else

b[i]=X[i];

X[i]=Y[i]=0;

}

}

void Sqrt(int len)

{

if(len==1)

{

sqr[0]=1;//本题被开方的多项式常数项为1

return;

}

Sqrt(len>>1);

Inverse(sqr,A,len);//求逆元

for(int i=0;i<(len>>1);i++)

B[i]=sqr[i];

for(int i=0;i<len;i++)

C[i]=g[i];

init(len\*2);

ntt(A,1);

ntt(B,1);

ntt(C,1);

for(int i=0;i<n;i++)

A[i]=(1ll\*C[i]+(LL)B[i]\*B[i])%p\*I2%p\*A[i]%p;//求平方根

ntt(A,0);//反演

for(int i=0;i<n;i++)

{

sqr[i]=A[i];

if(i>=len)

sqr[i]=0;

A[i]=B[i]=C[i]=0;

}

}

int main()

{

cin>>nn>>mm;

while(mx<=mm)

mx<<=1;

for(int i=1;i<=nn;i++)

{

scanf("%d",&by);

if(by<=mm)

g[by]=1;

}

for(int i=0;i<mx;i++)

if(g[i]) g[i]=p-4;//-4\*C(x)

g[0]=1;

Sqrt(mx);//

sqr[0]=(sqr[0]+1)%p;//

mmst(g,0);

Inverse(sqr,g,mx);//

for(int i=1;i<=mm;i++)

printf("%d\n",(g[i]+g[i])%p);//

return 0;

}